AD-A090 773 WASHINGTON UNIV SEATTLE DEPT OF COMPUTER SCIENCE F/G 9/2
ON DEVELOPING A THEORY OF DISTRIBUTED COMPUTING: SUMMARY OF CUR--ETC(U)
SEP 80 M J FISCHER N00014-80-C-0221
UNCLASSIFIED TR-80-09-03 NL

1 OF 1
AD A
090773

END
DATE
FILMED
11-80
DTIC

LEVEL II (12)

ON DEVELOPING A THEORY OF DISTRIBUTED COMPUTING:*
Summary of Current Research

Michael J. Fischer
Department of Computer Science, FR-35
University of Washington
Seattle, Washington 98195

Technical Report 80-09-03

September 1980

To be discussed at the Workshop on Fundamental Issues in Distributed Computing, December 15-17, 1980, Pala Mesa Resort, Fallbrook, California.

*This work was supported in part by the Office of Naval Research under Contracts N00014-80-C-0221 and N00014-79-C-0873 and by the National Science Foundation Grant No. MCS-77-02474.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 80-09-03	2. GOVT ACCESSION NO. AD-A090 773	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ON DEVELOPING A THEORY OF DISTRIBUTED COMPUTING: Summary of Current Research		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael J. Fischer, University of Washington		8. CONTRACT OR GRANT NUMBER(s) ONR Contracts: N00014-80-C-0221, N00014-79-C-0873; NSF Grant #MCS-77-02474.
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science, FR-35 University of Washington Seattle, WA 98195		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 049-456/30 Oct 79 (437)
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research 800 N. Quincy Street Arlington, VA 22217; National Science Found. Math. & Comp. Sciences Washington, D.C. 20550		12. REPORT DATE September 1980
		13. NUMBER OF PAGES 12
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research 800 N. Quincy Street Arlington, VA 22217 Attn: Dr. R. B. Grafton		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Distributed computing, concurrency, mathematical theory, complexity.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The paradigms of theory can be applied to the problems of concurrency and distribution to yield a rich mathematical theory which will provide a solid framework for practitioner and theoretician alike to use in gaining greater understanding of the exciting and important area of distributed computing. The author is currently engaged in an effort to develop such a theory. This paper summarizes work to date and indicates directions for further research.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102 LF 014 6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Prologue

I am currently involved in a research effort to apply theoretical tools to the rapidly emerging area of distributed computing. I believe that the paradigms of theory can be applied to the problems of concurrency and distribution to yield a rich mathematical theory which will provide a solid framework for practitioner and theoretician alike to use in gaining greater understanding of this exciting and important new component of computer science. The long-term goal is to develop a theory of distributed and concurrent computation analogous to the theory of sequential computation that has emerged over the past fifty years. Such an effort will require the involvement of a large segment of the theoretical research community. My goal here is to present a few examples of work to date which give the flavor of my work and to point out areas in which interaction with practitioners could be particularly beneficial.

1. The Elements of a Theory

By a theory, I mean the result of removing ambiguity and uncertainty in the statement of a problem so that precise, rigorous statements about it can be made and verified. Abstraction, the process of eliminating uninteresting detail, is an essential part of the process of constructing a theory, for it permits the problem to be reduced to a tractable and comprehensible size from which significant new insights can be obtained. The danger in studying abstractions is that certain essential elements may have been discarded along with the inessential, resulting in conclusions that are valid for the abstraction but nevertheless may not hold for the original, motivating, real-world problem. The only answer to this difficulty that I know of is to keep always aware of the problem, constantly question the assumptions being made, check the conclusions against reality, and study a variety of different abstractions.

The words "distributed computing" suggest many different new aspects of computer science introduced by the advent of networks of independent communicating processors, and different kinds of theories are likely to emerge from each. A few that we have investigated to date include problems of concurrency and synchronization, mechanisms and protocols for interprocessor communication, tools for performance measurement and analysis, effects of geometry and topology, and reliability considerations. By way of contrast, other important aspects which I have not addressed include programming issues such as naming and binding, nor have I looked at protection and security issues.

I give below some brief sketches of my work on a variety of these topics to illustrate the kinds of questions I have been asking and the sorts of results that can be obtained.

2. Synchronization Problems

My first work in this area concerned questions of synchronization of concurrent processes in environments in which keeping small the amount of information exhcanged by processes was an important consideration. In [PF], Gary Peterson and I looked into the amount of "communication space" that was required to solve the classical "critical section" problem for n processes which communicated using the "message board" model in which each process has a communication variable which only it can write and every other process can read. Even without reference to clocks or other timing considerations, we were able to exhibit a fair solution which used only a constant amount of communication space per process, independent of the total number of processes in the system.

In [BFJLP], we looked at the same problems in a much stronger (and less "distributed") model in which processes communicated via shared memory accessed through a general "test-and-set" instruction. Even with such a powerful model, we were able to prove non-trivial lower bounds on the rate at which communication space must grow as the number of processes to be synchronized increases, when various fairness constraints are imposed. In particular, $n/2$ values are required for any starvation-free solution for n processes, and $n+1$ are needed if in addition the solution has bounded-waiting. To show that these bounds are essentially best-possible, we constructed actual algorithms

that nearly achieve them. While the algorithms themselves may be of limited practical interest, they involve several novel programming "tricks" and internal protocols which may be useful in other contexts.

3. Models of Behavior

Virtually all of the work on the critical-section problem, our own included, suffered from the fact that the problem was stated in terms of particular desired internal properties of the computing system (e.g. mutual exclusion) rather than on observable behavior. It's nice to be able to quantify the cost of achieving mutual exclusion, but it's even nicer to be able to state a problem for which mutual exclusion is necessary. Looked at from another point of view, one does not want the statement of a computing problem to unduly constrain the system designer. For example, the specifications for an airline reservation system might state the numbers and locations of terminals, functional requirements, expected response times, etc., but it should not prescribe things like the number and locations of internal nodes in the network or its topology.

These considerations, together with the need for a formal model appropriate for performing time analyses, led Nancy Lynch and me to define an abstract notion of the behavior of a distributed system based only on observable input and outputs to the system and not on the internal workings of the system [LFa].

4. Timing Analysis

As in the case of sequential complexity theory, it makes sense to measure both "worst-case" running times and "expected" times under some

assumed probability distribution. The advantage of the former is that it is independent of difficult-to-defend probability assumptions, gives absolute bounds concerning performance (and hence correctness) in a real-time environment, and is often far more tractable mathematically. The latter of course may reflect the observable realities far better.

In the case of distributed algorithms, however, it is not always so clear just what to measure, for non-terminating programs are the rule and there may be a trade-off between performance measures such as throughput and response time. We suggest some examples of appropriate timing measures for the particularly simple arbiter problem in [FLa], and we expand on those ideas in the revised version of that paper [LFb]. These are worst-case kinds of analyses. Nancy Lynch, Ed Lazowska, Pat Jacobson and I are currently trying to perform an expected-time analysis of these same systems for comparison, using queuing theory techniques.

5. Reliability and Fault-Tolerance

In [FLBB], we looked at the seemingly minor generalization of the critical section problem to permit a maximum of some number ℓ of processes to be simultaneously in their critical sections but not more. Obvious generalizations of 1-critical section solutions become unattractive when we require in addition that the system "tolerate" the failure of a limited number of processes. The kind of fault we consider is a process simply ceasing to take further steps, but it does not announce this fact to the other processes in any way. Since we do not include clocks in our model, "time-outs" are impossible, so there is no way for one process to tell whether another has

failed or is just running very slowly. Our main results are that there is an algorithm for solving this problem which tolerates limited process failure, satisfies a suitably-generalized notion of "bounded-waiting", and uses only $O(n)$ values of shared communication memory. For FIFO fairness property, we need only $O(n (\log n)^c)$ values of shared memory. Again we hope the principles used in the design of these algorithms, if not the algorithms themselves, will be exportable to other problems of reliable concurrent computation.

Michael Rabin, Udi Manber and I in some current research are investigating another problem in which the key technical difficulty is to achieve reliable operation of the non-failing portion of the system. Here our model is a large number of independent processes all simultaneously searching a directed graph represented in list-structured memory. Because of memory constraints, the problem is not for processes to cooperate with each other but simply for them to stay out of each other's way. The results of our efforts so far are a surprisingly non-trivial algorithm for solving this problem in small space and a new kind of "locking" protocol.

A third piece of current work, this time joint with Nancy Lynch and Leslie Lamport, concerns the paper of Pease, Shostak, and Lamport [PSL] on coping with "malicious failure" in which the failed processes might continue to take steps but produce erroneous and unpredictable results. The algorithm of [PSL] for achieving agreement in the presence of such faults works in stages or "rounds" in which each process exchanges information with each other. To protect against a maximum of k faulty processes, the algorithm requires $k+1$ rounds (and hence time at least $k+1$). Although a faster

algorithm would be very desirable, we can show that no algorithm can solve this problem in fewer than k+1 rounds. We think similar results can also be obtained for the cases where messages are non-forgable and where the processes are asynchronous.

6. Resource-Placement Problems

The allocation of resources to processes in a distributed system is particularly challenging and seems to underlie many other problems of distributed computation. We imagine a fixed number of resources or "tickets" for which requests may originate at various points in the system. The basic property of a ticket system is that it never grant more requests than there are tickets available. Other possibly desirable properties are that a request not get denied when tickets are available elsewhere in the system, and that the system be able to accept the return of tickets.

In joint work with Nancy Lynch and Nancy Griffeth, we began by investigating a number of ticket algorithms for networks of various topologies but had difficulty in comparing them. This led us to abstract further to a very simple problem: assume the network is a complete binary tree, that requests originate at randomly chosen leaves, and that requests are matched to tickets in an optimal way. The problem is then to find what initial placement of tickets on the nodes of the tree leads to the least expected distance between a request and the ticket which satisfies that request. Rather surprisingly, we found that the expected distance for the optimal placement is constant as long as the number of tickets (and requests) is at least proportional to the number of leaves. This suggests that, at least in certain circumstances,

algorithms which dynamically move the tickets around the system can at best achieve only limited improvements over that obtainable by static placement algorithms.

7. Conclusion

The work described above spans a considerable number of topics of practical importance in distributed computing. I feel it essential to learn more about the practical issues in order to form more meaningful and relevant abstractions. At the same time, I feel it valuable to share the insights I have already attained through these studies with others in the field. This workshop seems an ideal forum for such interchange.

Acknowledgement

Nancy Lynch has worked closely with me on much of the work described above and has contributed greatly to the overall direction of the research as well as to the particular research topics. I am indebted to her for her creativity, insights, and just plain hard work that made this all possible. I am also grateful to my several coauthors and coworkers, whose contributions are immeasurable.

References

[BFJLP] J. E. Burns, M. J. Fischer, P. Jackson, N. A. Lynch, and G. L. Peterson, "Shared Data Requirements for Implementation of Mutual Exclusion Using a Test-and-Set Primitive," Proc. 1978 International Conf. on Parallel Processing, IEEE Catalog No. 78CH1321-9C (August 1978), 79-87.

[FGL] M. J. Fischer, N. D. Griffeth and N. A. Lynch, "Optimal Resource Placement in a Distributed System," University of Washington Technical Report 80-08-03 (August 1980).

[FLBB] M. J. Fischer, N. A. Lynch, J. E. Burns and A. Borodin, "Resource Allocation with Immunity to Limited Process Failure," 20th IEEE Symposium on Foundations of Computer Science (October 1979), 234-254.

[LFa] N. A. Lynch and M. J. Fischer, "On Describing the Behavior and Implementation of Distributed Systems," (preliminary report) in Semantics of Concurrent Computation, ed. G. Kahn, Vol. 70 of Lecture Notes in Computer Science series, Springer-Verlag, 1979, 147-171.

[LFb] N. A. Lynch and M. J. Fischer, "On Describing the Behavior and Implementation of Distributed Systems," to appear in Theoretical Computer Science.

[PF] G. L. Peterson and M. J. Fischer, "Economical Solutions for the Critical Section Problem in a Distributed System," Proc. Ninth ACM Symp. on Theory of Computing (1977), 91-97.

[PSL] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreements in the Presence of Faults," J. ACM 27, 2 (April 1980), 228-234.

DISTRIBUTION LIST

Office of Naval Research Contract N00014-80-C-0221
Michael J. Fischer, Principal Investigator

Defense Documentation Center
Cameron Station
Alexandria, VA 22314
(12 copies)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

- Dr. R. B. Grafton, Scientific Officer (1 copy)
- Information Systems Program (437) (2 copies)
- Code 200 (1 copy)
- Code 455 (1 copy)
- Code 458 (1 copy)

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106
(1 copy)

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375
(6 copies)

Office of Naval Research
Resident Representative
University of Washington, JD-27
422 University District Building
1107 NE 45th Street
(1 copy)

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380
(1 copy)

Naval Ocean Systems Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152
(1 copy)

Mr. E. H. Gleissner
Naval Ship Research and Development Center
Computation and Mathematics Dept.
Bethesda, MD 20084
(1 copy)

Captain Grace M. Hooper (008)
Naval Data Automation Command
Washington Navy Yard
Building 166
Washington, D.C. 20374
(1 copy)

Defense Advanced Research Projects Agency
Attn: Program Management/MIS
1400 Wilson Boulevard
Arlington, VA 22209
(3 copies)

Professor Nancy A. Lynch
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332

Professor Philip Enslow
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, IL 60605